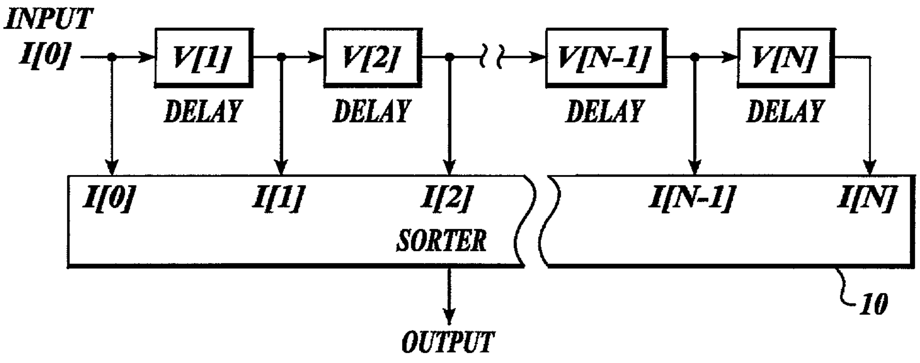

Fig. 1. (PRIOR ART)

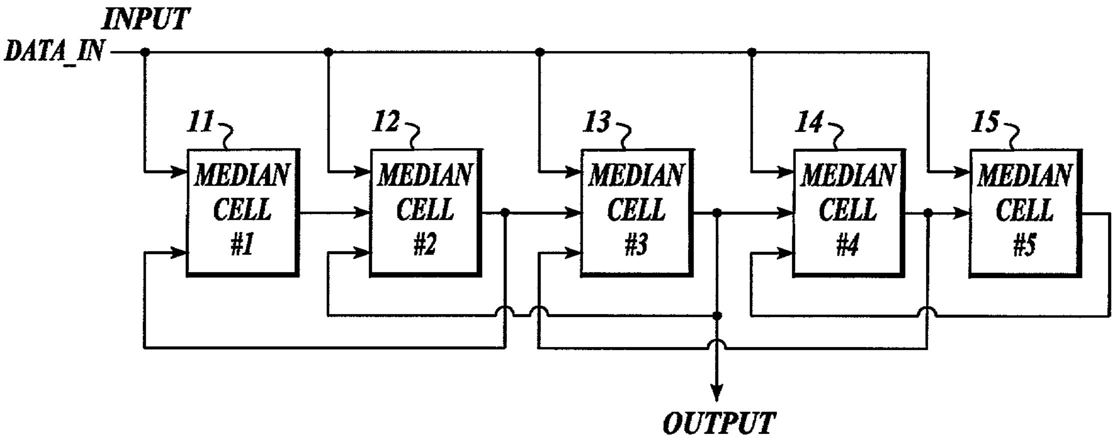

Fig. 2. (PRIOR ART)

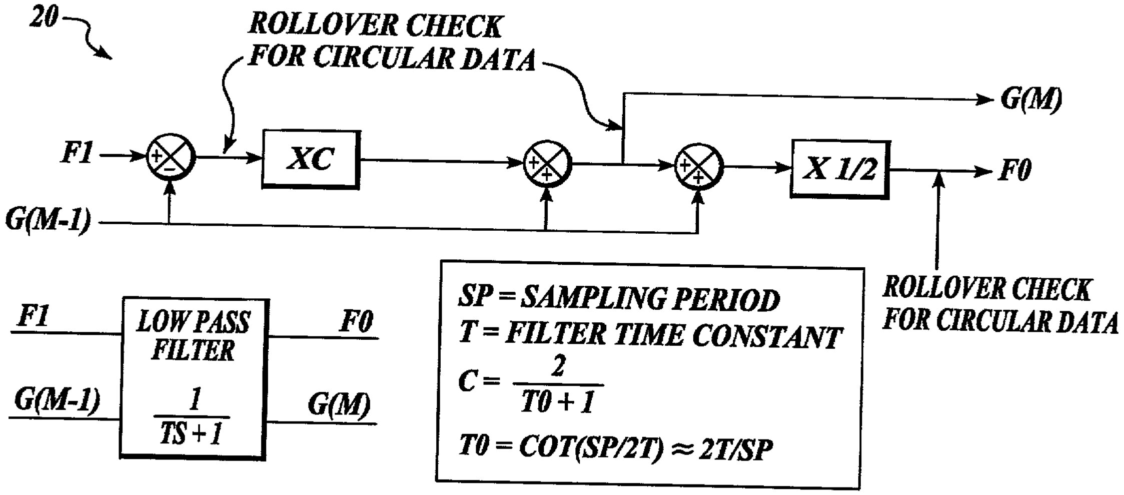

Fig. 3. (PRIOR ART)

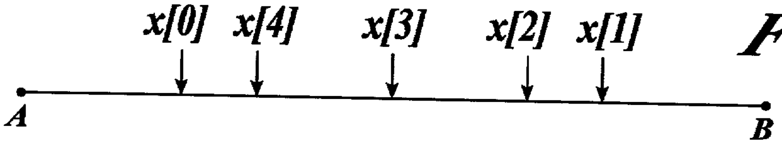

Fig. 4.

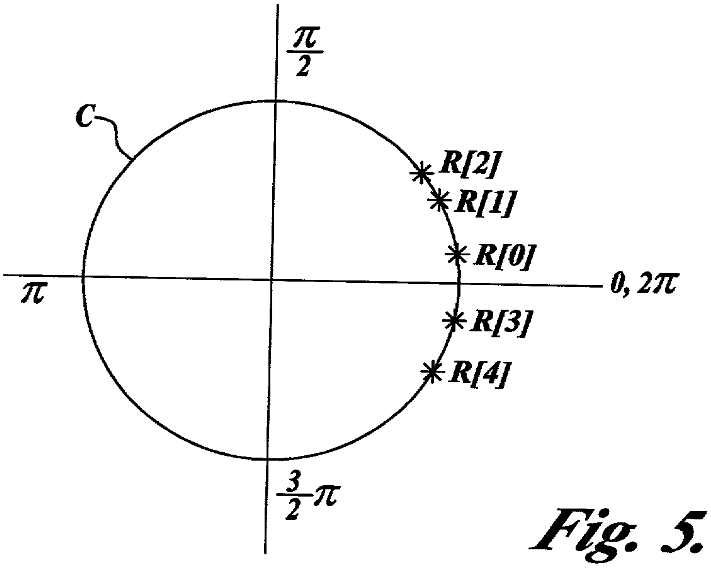

Fig. 5.

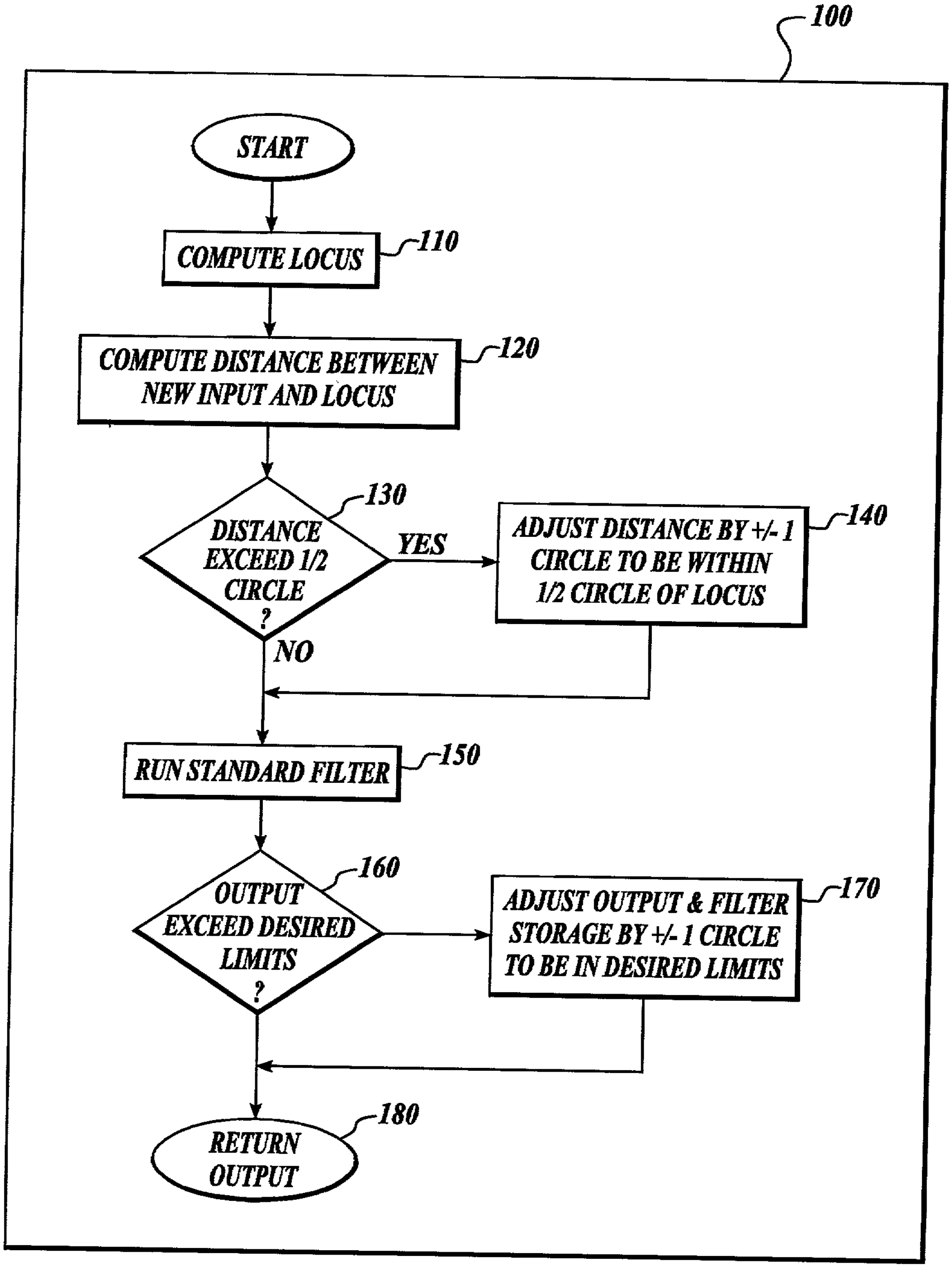
100
START
COMPUTE LOCUS
110
COMPUTE DISTANCE BETWEEN NEW INPUT AND LOCUS
120
DISTANCE EXCEED 1/2 CIRCLE ?
130
YES
ADJUST DISTANCE BY +/- 1 CIRCLE TO BE WITHIN 1/2 CIRCLE OF LOCUS
140
NO
RUN STANDARD FILTER
150
OUTPUT EXCEED DESIRED LIMITS ?
160
ADJUST OUTPUT & FILTER STORAGE BY +/- 1 CIRCLE TO BE IN DESIRED LIMITS
170
RETURN OUTPUT
180

Fig. 6.

```
//*************************************************************
// CLASS CircMedian3Filter (derived from Median3Filter)
//*************************************************************

// rInp = value to be filtered, returned Real is filtered value
Real CircMedian3Filter::Filter(Real rInp)
{
Real rAvg = (arF[0] + arF[l]) / 2;

   if ((rInp - rAvg) > rHALFCIRCLE)
       rInp -= rFULLCIRCLE;
   else if ((rInp - rAvg) < - rHALFCIRCLE)
      rInp += rFULLCIRCLE;

Real r =  Median3Filter::Filter(rInp);

   // normalize out any circular adjustments
    if (r > rFULLCIRCLE)
    {
       arF[0] -= rFULLCIRCLE;
       arF[l]  -= rFULLCIRCLE;
       r -= rFULLCIRCLE;
    }
   else if (r < -rHALFCIRCLE)
    {
       arF[0] += rFULLCIRCLE;
       arF[l] += rFULLCIRCLE;
       r += rFULLCIRCLE;
    }

   return (r); // return median value
} // CircMedian3Filter::Filter
```

Fig. 7.

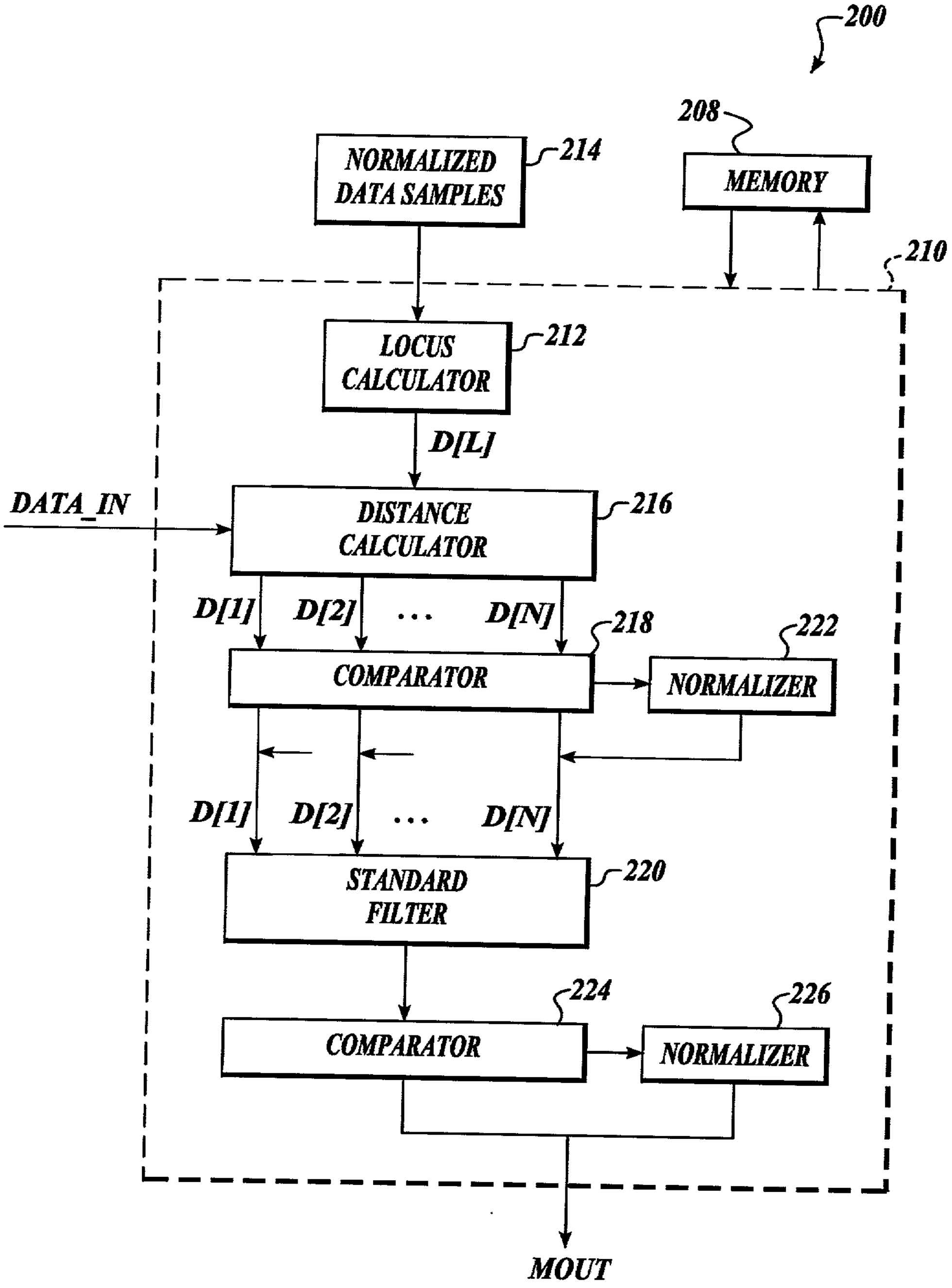
200
208
MEMORY
NORMALIZED DATA SAMPLES
214
210
LOCUS CALCULATOR
212
D[L]
DATA_IN
DISTANCE CALCULATOR
216
D[1]
D[2]
...
D[N]
218
COMPARATOR
222
NORMALIZER
D[1]
D[2]
...
D[N]
STANDARD FILTER
220
224
COMPARATOR
226
NORMALIZER
MOUT

Fig. 8.

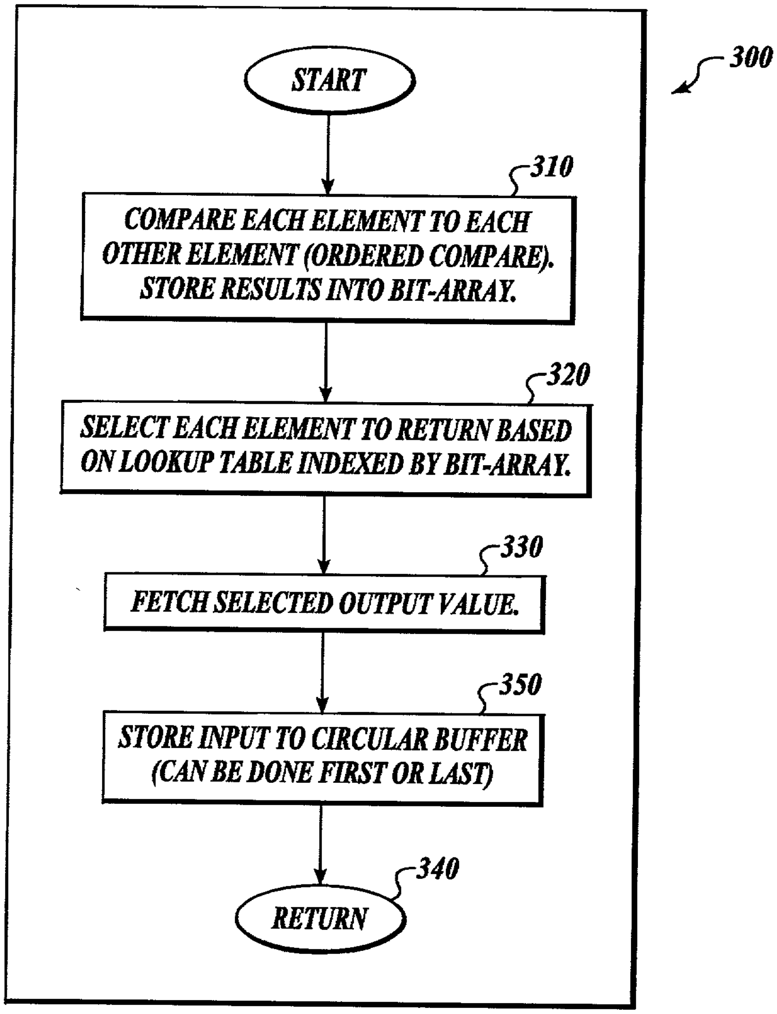
300
START
310
COMPARE EACH ELEMENT TO EACH OTHER ELEMENT (ORDERED COMPARE). STORE RESULTS INTO BIT-ARRAY.
320
SELECT EACH ELEMENT TO RETURN BASED ON LOOKUP TABLE INDEXED BY BIT-ARRAY.
330
FETCH SELECTED OUTPUT VALUE.
350
STORE INPUT TO CIRCULAR BUFFER (CAN BE DONE FIRST OR LAST)
340
RETURN

Fig. 9.

```
//**************************************************************
// CLASS Median3Filter
//************************************************************
Median3Filter : : Median3Filter(void)
{
SetTo(0);
}   // Median3Filter : : Median3Filter
//************************************************************
// rInp = value to be filtered, returned Real is filtered
value
Real median3Filter : : Filter(Real rInp)
{
int iCase = 0;
    if (arF[0] > arF[l])
    iCase | = 1;
    if (arF[l] > rInp)
    iCase | = 2;
    if (rInp > arF[0])
    iCase | = 4;
Real r = 0;
    switch (iCase)
    {
        case 2:  // 010:  0 <= l,  1 > 2,  2 <= 0  (1 > 0 > 2)
        case 5:  // 101:  0 > 1,  1 <= 2,  2 > 0  (2 > 0 > 1)
            r = arF[0];
            break;
        case 0:  // 000: 0 <= l,  1 <= 2,  2 <= 0  (0 = 1 = 2)
        case 3:  // 0ll:  0 > 1,  1 > 2,  2 <= 0  (0 > 1 > 2)
        case 4:  // 100: 0 <= l,  1 <= 2,  2 > 0  (2 > 1 > 0)
            r = arF[l];
            break;
        case 1:  // 001: 0 > 1,  1 <= 2,  2 <= 0  (0 > 2 > 1)
        case 6:  // 110:  0 <= l,  1 > 2,  2 > 0  (1 > 2 > 0)
            r = rInp;
            break;
//      case 7:  // 111:  0 > 1,   1 > 2,   2 > 0   (illogical)
//      default: //  MORE THAN 3 BITS SET   (N/A)
    }
    arF[idx] = rInp;
    idx ^= 1;
    return (r); // return median value
}   // Median3Filter : : Filter
//************************************************************
// rInp = value filter is preset to
void Median3Filter : : SetTo(Real rInp)
{
    arF[0] = arF[l] = rInp;
    idx = 0;
} // Median3Filter : : SetTo
```

Fig. 10.

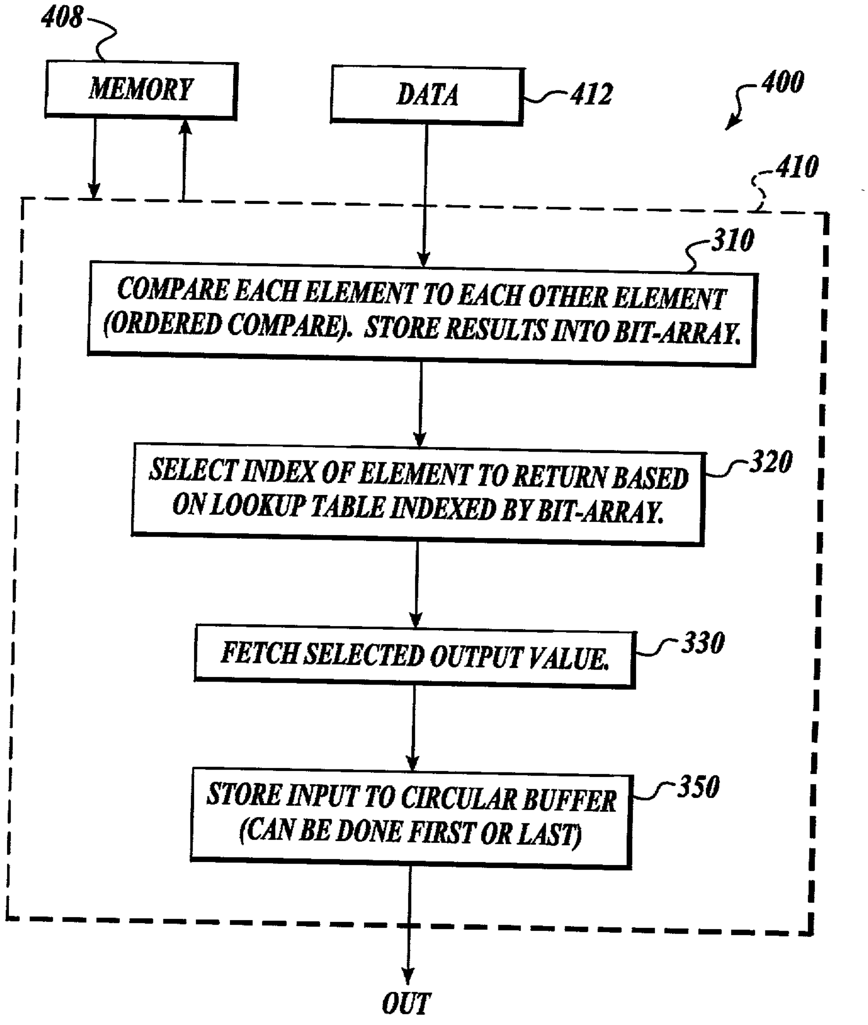
408
MEMORY
DATA
412
400
410
310
COMPARE EACH ELEMENT TO EACH OTHER ELEMENT (ORDERED COMPARE). STORE RESULTS INTO BIT-ARRAY.
320
SELECT INDEX OF ELEMENT TO RETURN BASED ON LOOKUP TABLE INDEXED BY BIT-ARRAY.
330
FETCH SELECTED OUTPUT VALUE.
350
STORE INPUT TO CIRCULAR BUFFER (CAN BE DONE FIRST OR LAST)
OUT

Fig. 11.